## 6.Library Management System

### 1. Understand Search Algorithms

**Linear Search:**

- **Description:** Linear search involves iterating through each element in the list until the desired element is found or the list ends.
- **Steps:**
    1. Start from the first element.
    2. Compare each element with the target value.
    3. If a match is found, return the index of the element.
    4. If the end of the list is reached without finding the target, return -1 or indicate that the target is not present.

**Time Complexity:** O(n), where n is the number of elements in the list.

**Advantages:**

- Simple and straightforward.
- No need for the list to be sorted.
- Works well for small datasets.

**Disadvantages:**

- Inefficient for large datasets due to O(n) time complexity.

**Binary Search:**

- **Description:** Binary search is a more efficient algorithm for finding an element in a sorted list by repeatedly dividing the search interval in half.
- **Steps:**
    1. Start with the entire list.
    2. Find the middle element of the list.
    3. Compare the middle element with the target value.
    4. If the middle element is equal to the target, return the index.
    5. If the target is less than the middle element, repeat the search on the left half.
    6. If the target is greater than the middle element, repeat the search on the right half.
    7. Continue until the target is found or the interval is empty.

**Time Complexity:** O(log n), where n is the number of elements in the list.

**Advantages:**

- Much faster than linear search for large datasets.
- Efficient for sorted lists.

**Disadvantages:**

- Requires the list to be sorted.
- More complex to implement than linear search.

**2. Setup**

```
class Book {
    private int bookId;
    private String title;
    private String author;

    public Book(int bookId, String title, String author) {
        this.bookId = bookId;
        this.title = title;
        this.author = author;
    }

    public int getBookId() {
        return bookId;
    }

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }
```

```
    @Override
    public String toString() {
        return "Book ID: " + bookId + ", Title: " + title + ", Author: " + author;
    }
}
```

**3. Implementation**

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.Scanner;
public class LibraryManagementSystem {


    public static Book linearSearch(List<Book> books, String title) {
        for (Book book : books) {
            if (book.getTitle().equalsIgnoreCase(title)) {
                return book;
            }
        }
        return null;
    }


    public static Book binarySearch(List<Book> books, String title) {
        int left = 0;
        int right = books.size() - 1;
```

```
    while (left <= right) {
        int mid = left + (right - left) / 2;
        int comparison = books.get(mid).getTitle().compareToIgnoreCase(title);

        if (comparison == 0) {
            return books.get(mid);
        } else if (comparison < 0) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return null;
}

public static void main(String[] args) {
    List<Book> books = new ArrayList<>();
    books.add(new Book(1, "The Great Gatsby", "F. Scott Fitzgerald"));
    books.add(new Book(2, "To Kill a Mockingbird", "Harper Lee"));
    books.add(new Book(3, "1984", "George Orwell"));
    books.add(new Book(4, "Pride and Prejudice", "Jane Austen"));
    books.add(new Book(5, "Moby Dick", "Herman Melville"));
    Collections.sort(books, Comparator.comparing(Book::getTitle));

    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the title of the book to search (using linear search): ");
    String title = scanner.nextLine();

    Book foundBookLinear = linearSearch(books, title);
    if (foundBookLinear != null) {
```

```
            System.out.println("Found using linear search: " + foundBookLinear);
        } else {
            System.out.println("Book not found using linear search.");
        }

        System.out.print("Enter the title of the book to search (using binary search): ");
        title = scanner.nextLine();

        Book foundBookBinary = binarySearch(books, title);
        if (foundBookBinary != null) {
            System.out.println("Found using binary search: " + foundBookBinary);
        } else {
            System.out.println("Book not found using binary search.");
        }

    }
}
```

**4. Analysis**

**Time Complexity Comparison:**

- **Linear Search:** O(n)
  - Searches each element sequentially.
  - Effective for small or unsorted datasets.
  - Simple to implement but becomes inefficient for large datasets.
- **Binary Search:** O(log n)
  - Requires the list to be sorted.
  - Divides the search interval in half each time.
  - Much more efficient for large datasets due to logarithmic time complexity.

**When to Use Each Algorithm:**

- **Linear Search:**

- Use when the dataset is small or unsorted.
- Useful when simplicity is needed and the overhead of sorting is not justified.
- Effective when the frequency of search operations is low compared to the frequency of insertions and deletions.

- **Binary Search:**
  - Use when the dataset is large and sorted.
  - Preferred when search operations are frequent and the overhead of maintaining a sorted list is justified.
  - Efficient for static datasets where the data doesn't change often, allowing the list to remain sorted.